HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Project

Thực hành kiến trúc máy tính

Nhóm 1

Phạm Việt Anh – 20225599
Mạch Ngọc Đức Anh - 20225595

ONE LOVE. ONE FUTURE.

Nội dung

Project 5: Hàm cấp phát bộ nhớ malloc()

Thực hiện: Mạch Ngọc Đức Anh

Project 6: Chương trình kiểm tra cú pháp lệnh MIPS

Thực hiện: Phạm Việt Anh

Yêu cầu của bài

Hàm cấp phát bộ nhớ malloc

1) Việc cấp phát bộ nhớ kiểu word/mảng word có 1 lỗi, đó là chưa bảo đảm qui tắc địa chỉ của kiểu word phải chia hết cho 4. Hãy khắc phục lỗi này.
2) Viết hàm lấy giá trị Word /Byte của biến con trỏ (tương tự như **CharPtr*, **BytePtr*, **WordPtr*)
3) Viết hàm lấy địa chỉ biến con trỏ (tương tự như *&CharPtr*, *&BytePtr*, *&WordPtr*)
4) Viết hàm thực hiện copy 2 con trỏ xâu kí tự (Xem ví dụ về *CharPtr*)
5) Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát cho các biến động
6) Hãy viết hàm Malloc2 để cấp phát cho mảng 2 chiều kiểu *.word* với tham số vào gồm:
a. Địa chỉ đầu của mảng
b. Số dòng
c. Số cột
7) Tiếp theo câu 6, hãy viết 2 hàm *GetArray[i][j]* và *SetArray[i][j]* để lấy/thiết lập giá trị cho phần tử
ở dòng *i* cột *j* của mảng.

Tạo các hàm theo yêu cầu

Khai báo các biến liên quan đến bộ nhớ kernel

```
.kdata
        #Bien chua dia chi dau tien cua vung nho con trong
        Sys_TheTopOfFree: .word 1
        #Vung khong gian tu do, dung de cap bo nho cho cac bien con tro
        Sys_MyFreeSpace
```

Tạo biến đếm bộ nhớ đã cấp phát

```
.text
        #Khoi tao bien dem bo nho dung thuc
          li $t0, 0
          #Khoi tao bien dem bo nho dung sau chuan hoa
          li $t3, 0
```

Hàm khởi tạo bộ nhớ để cấp phát động

```
SysInitMem:
        la $t9, Sys_TheTopOfFree   #Lay con tro chua dau tien con trong, khoi tao
        la $t7, Sys_MyFreeSpace.    #Lay dia chi dau tien con trong, khoi tao
        sw $t7, 0($t9) #Luu lai
        jr $ra
```

Tạo các hàm theo yêu cầu

Hàm malloc

```
malloc:
        la $t9, Sys_TheTopOfFree
        lw $t8, 0($t9) #Lay dia chi dau tien con trong
        sub $t3,$t3,$t8
        add $s2, $zero, $a0 # luu gia tri dia chi con tro
vao bien tam thoi $s2
        li $v0, 34
        add $a0, $t8, $zero
        syscall

        li $v0, 4
        la $a0, Space
        syscall
        checkDivide4:
                li $t4, 4
                div $t8, $t4
                mfhi $t5
                beq $t5, $zero, done
                addi $t8, $t8,1
                j checkDivide4
        done:
        add $t3, $t8, $t3
         li $v0, 34
        add $a0, $t8, $zero
        syscall

        li $v0, 4
        la $a0, newline
        syscall

        add $a0, $zero, $s2 #gan lai gia tri bien tam thoi vao lai $a0
        sw $t8, 0($a0) #Cat dia chi do vao bien con tro
        addi $v0, $t8, 0 #Dong thoi la ket qua tra ve cua ham
        mul $t7, $a1, $a2 #Tinh kich thuoc cua mang can cap nhat
        add $t0, $t0, $t7
        add $t3, $t3, $t7
        add $t6, $t8, $t7 #Tinh dia chi dau tien con trong
        sw $t6, 0($t9) #Luu tro lai dia chi dau tien do vao bien
Sys_TheTopOfFree
        jr $ra
```

Tạo các hàm theo yêu cầu

Hàm lấy giá trị của biến con trỏ

```
getValue:
        lw $v0, 0($a0)
        jr $
```

Hàm lấy địa chỉ của biến con trỏ

```
getAddress:
        add $v0, $zero, $a0
        jr $ra
```

Tạo các hàm theo yêu cầu

Hàm copy 2 con trỏ xâu ký tự

```
CopyPointer:
        sw $a0, 0($a1)
        jr $ra
```

Hàm tính toàn bộ lượng bộ nhớ đã cấp phát cho các biến động

```
#Khong tinh phan nhay coc dia chi
    CalculateMemory:
        add $a0, $zero, $t0
        jr $ra
#Khong tinh phan nhay coc dia chi
    CalculateMemory2:
        add $a0, $t3, $zero
        jr $ra
```

Tạo các hàm theo yêu cầu

Hàm Malloc2 để cấp phát cho mảng 2 chiều

```
malloc2:
        la $t9, Sys_TheTopOfFree
        lw $t8, 0($t9) #Lay dia chi dau tien con trong
        sub $t3,$t3, $t8
        checkDevide4_2:
                li $t4, 4
                div $t8, $t4
                mfhi $t5
                beq $t5, $zero, done_2
                addi $t8, $t8,1
                j checkDevide4_2
        done_2:
                add $t3,$t3,$t8
        sw $t8, 0($a0) #Cat dia chi do vao bien con tro
        addi $v0, $t8, 0 #Dong thoi la ket qua tra ve cua ham
        mul $t7, $a1, $a2 #Tinh so luong phan tu
        mul $t7, $t7, 4 #Tinh kich thuoc bo nho can cung cap
        add $t0, $t0, $t7
        add $t3, $t3, $t7
        add $t6, $t8, $t7 #Tinh dia chi dau tien con trong
        sw $t6, 0($t9) #Luu tro lai dia chi dau tien do vao bien
Sys_TheTopOfFree
        jr $ra
```

Tạo các hàm theo yêu cầu

Hàm lấy giá trị phần tử ở dòng i cột j

```
GetArrayAt:
        la $t1, MaxDong
        lw $t1,0($t1)
        la $t2, MaxCot
        lw $t2, 0($t2)
        mul $t7, $a1, $t2
        add $t7, $t7, $a2
        mul $t7, $t7, 4
        add $a0, $a0, $t7
        lw $v0, 0($a0)
        jr $ra
```

Hàm thiết lập giá trị phần tử ở dòng i cột j

```
SetArrayAt:
        la $t1, MaxDong
        lw $t1,0($t1)
        la $t2, MaxCot
        lw $t2, 0($t2)
        mul $t7, $a1, $t2
        add $t7, $t7, $a2
        mul $t7, $t7, 4
        add $a0, $a0, $t7
        sw $v0, 0($a0)
        jr $ra
```

Khởi tạo câu dẫn và con trỏ

```
.data
        CharPtr: .word 0 # Bien con tro, tro toi kieu asciiz
        BytePtr: .word 0 # Bien con tro, tro toi kieu Byte
        WordPtr: .word 0 # Bien con tro, tro toi mang kieu Word
        CpyCharPtr: .word 0
        ArrayWordPtr: .word 0
        MaxDong: .word 5
        MaxCot: .word 5
        Space: .asciiz " -> "
        newline: .asciiz "\n\n"
        Message11: .asciiz "Dia chi o nho CharPtr da chuan hoa la: "
        Message12: .asciiz "Dia chi o nho BytePtr da chuan hoa la: "
        Message13: .asciiz "Dia chi o nho WordPtr da chuan hoa la: "
        Message21: .asciiz " Gia tri cua con tro CharPtr la: "
        Message22: .asciiz " Gia tri cua con tro BytePtr la: "
        Message23: .asciiz " Gia tri cua con tro WordPtr la: "
        Message31: .asciiz " Dia chi cua bien con tro la: "
        Message32: .asciiz " Dia chi cua bien con tro la: "
        Message33: .asciiz " Dia chi cua bien con tro la: "
        Message4: .asciiz "Q4: Dia chi cua con tro va con tro copy la: "
        Message5: .asciiz "Q5: Luong bo nho da cap phat la: "
        Message6: .asciiz "Q6: Luong bo nho sau khi su dung malloc 2 (thuc dung vs chuan hoa): "
        Message71: .asciiz "Q7.1: SetArray[3][3], Set gia tri 1007 cho phan tu vi tri [3][3] cua mang co dia chi la: "
        Message72: .asciiz "Q7.2: GetArray[3][3], Lay gia tri phan tu vi tri [3][3] cua mang: "
        exitMess: .asciiz "The index is out of range”
        endmess: .asciiz "end program "
```

Ví dụ minh hoạ sử dung hàm

Cấp phát cho biến con trỏ và đưa ra giá trị , địa chỉ con trỏ charptr

```
# Cap phat cho bien con tro, gom 3 phan tu, moi phan tu 1 byte
        #---------------------------------------
        la $a0, CharPtr
        addi $a1, $zero, 3
        addi $a2, $zero, 1
        li $v0, 4
        la $a0, Message11
        syscall
        jal malloc
        #---------------------------------------
        li $v0, 4
        la $a0, Message21
        syscall

        la $a0, CharPtr
        jal getValue
        add $a0, $v0, $zero
        li $v0, 1
        syscall
        li $v0, 4
        la $a0, newline
        syscall
        li $v0, 4
        la $a0, Message31
        syscall
        la $a0, CharPtr
        jal getAddress
        add $a0, $v0, $zero
        li $v0, 34
        syscall
        li $v0, 4
        la $a0, newline
        syscall
```

Ví dụ minh hoạ sử dung hàm

Cấp phát cho biến con trỏ và đưa ra giá trị , địa chỉ con trỏ byteptr

```
# Cap phat cho bien con tro, gom 6 phan tu, moi phan tu 1 byte
        la $a0, BytePtr
        addi $a1, $zero, 6
        addi $a2, $zero, 1
        li $v0, 100
        sw $v0, 0($a0)
        li $v0, 4
        la $a0, Message12
        syscall
        jal malloc

        li $v0, 4
        la $a0, Message22
        syscall

        la $a0, BytePtr
        jal getValue
        add $a0, $v0, $zero
        li $v0, 1
        syscall

        li $v0, 4
        la $a0, newline
        syscall

        li $v0, 4
        la $a0, Message32
        syscall

        la $a0, BytePtr
        jal getAddress
        add $a0, $v0, $zero
        li $v0, 34
        syscall

        li $v0, 4
        la $a0, newline
        syscall
```

Ví dụ minh hoạ sử dung hàm

Cấp phát cho biến con trỏ và đưa ra giá trị , địa chỉ con trỏ wordptr

```
# Cap phat cho bien con tro, gom 5 phan tu, moi phan tu 4 byte
        #-------------------------------------
        la $a0, WordPtr
        addi $a1, $zero, 5
        addi $a2, $zero, 4

        li $v0, 4
        la $a0, Message13
        syscall
        jal malloc

        li $v0, 4
        la $a0, Message23
        syscall

        la $a0, WordPtr
        jal getValue
        add $a0, $v0, $zero
        li $v0, 1
        syscall

        li $v0, 4
        la $a0, newline
        syscall

        li $v0, 4
        la $a0, Message33
        syscall

        la $a0, WordPtr
        jal getAddress
        add $a0, $v0, $zero
        li $v0, 34
        syscall

        li $v0, 4
        la $a0, newline
        syscall
```

Ví dụ minh hoạ

Copy 2 con trỏ

```
# Question 4: Copy hai con tro
        li $v0, 4
        la $a0, Message4
        syscall

        la $a0, CharPtr
        li $v0, 34
        syscall

        li $v0, 4
        la $a0, Space
        syscall

        la $a0, CharPtr
        la $a1, CpyCharPtr
        jal CopyPointer
        add $a1,$a1,$zero
        li $v0, 34
        syscall

        li $v0, 4
        la $a0, newline
        syscall
```

Ví dụ minh hoạ

Tính tổng bộ nhớ sử dụng

```
#Question 5: Tinh tong bo nho su dung
        li $v0, 4
        la $a0, Message5
        syscall

        jal CalculateMemory
        li $v0,1
        syscall
        li $v0, 4
        la $a0, Space
        syscall
        jal CalculateMemory2
        li $v0,1
        syscall

        li $v0, 4
        la $a0, newline
        syscall
```

Ví dụ minh hoạ

Cấp phát cho mảng con trỏ 2 chiều và tính tổng bộ nhớ sử dụng

```
        # Cap phat cho mang con tro word 2 chieu,
gom 5 dong, 5 cot moi phan tu 4 byte.
        #-------------------------------------

        la $a0, ArrayWordPtr
        la $a1, MaxDong
        lw $a1,0($a1)
        la $a2, MaxCot
        lw $a2, 0($a2)
        jal malloc2
```

```
        li $v0, 4
        la $a0, Message6
        syscall

        jal CalculateMemory
        li $v0,1
        syscall
        li $v0, 4
        la $a0, Space
        syscall
        jal CalculateMemory2
        li $v0,1
        syscall
```

Ví dụ minh hoạ

Lấy và thiết lập phần tử ở dòng i cột j

```
li $v0, 4
  la $a0, Message71
  syscall

  li $v0, 1007
  la $a0, ArrayWordPtr
  li $a1,3
  la $s0, MaxDong
  lw $s0, 0($s0)
  bge $a1, $s0, exit
  li $a2, 3
  la $s0, MaxCot
  lw $s0, 0($s0)
  bge $a2, $s0, exit
  jal SetArrayAt
  li $v0, 34
  syscall
```

```
li $v0, 4
  la $a0, Message72
  syscall

  la $a0, ArrayWordPtr
  li $a1,3
  la $s0, MaxDong
  lw $s0, 0($s0)
  bge $a1, $s0, exit
  li $a2, 3
  la $s0, MaxCot
  lw $s0, 0($s0)
  bge $a2, $s0, exit
  jal GetArrayAt
  add $a0,$v0,$zero
  li $v0, 1
  syscall
```

Kết quả mô phỏng

```
Dia chi o nho CharPtr da chuan hoa la: 0x90000004 -> 0x90000004
 Gia tri cua con tro CharPtr la: 0
 Dia chi cua bien con tro la: 0x10010000
Dia chi o nho BytePtr da chuan hoa la: 0x90000007 -> 0x90000008
 Gia tri cua con tro BytePtr la: 100
 Dia chi cua bien con tro la: 0x10010004
Dia chi o nho WordPtr da chuan hoa la: 0x9000000e -> 0x90000010
 Gia tri cua con tro WordPtr la: 0
 Dia chi cua bien con tro la: 0x10010008
Q4: Dia chi cua con tro va con tro copy la: 0x10010000 -> 0x10010000
Q5: Luong bo nho da cap phat la: 29 -> 32
Q6: Luong bo nho sau khi su dung malloc 2 (thuc dung vs chuan hoa): 129 -> 132
Q7.1: SetArray[3][3], Set gia tri 1007 cho phan tu vi tri [3][3] cua mang co dia chi la: 0x10010058
Q7.2: GetArray[3][3], Lay gia tri phan tu vi tri [3][3] cua mang: 1007
end program
-- program is finished running (dropped off bottom) --
```

Yêu cầu đề bài

Trình biên dịch của bộ xử lý MIPS sẽ tiến hành kiểm tra cú pháp các lệnh hợp ngữ trong mã nguồn, xem có phù hợp về cú pháp hay không, rồi mới tiến hành dịch các lệnh ra mã máy. Hãy viết một chương trình kiểm tra cú pháp của 1 lệnh hợp ngữ MIPS bất kì (không làm với giả lệnh) như sau:

- Nhập vào từ bàn phím một dòng lệnh hợp ngữ. Ví dụ: beq s1,31,t4

-Kiểm tra xem mã opcode có đúng hay không? Trong ví dụ trên, opcode là beq là hợp lệ thì hiển thị thông báo “opcode: beq, hợp lệ”

-Kiểm tra xem tên các toán hạng phía sau có hợp lệ hay không? Trong ví dụ trên, toán hạng s1 là hợp lệ, 31 là không hợp lệ, t4 thì khỏi phải kiểm tra nữa vì toán hạng trước đã bị sai rồi.

-Cho biết lệnh hợp ngữ đó thuộc dạng lệnh nào (R, I, J) và cần bao nhiêu chu kì thì mới thực hiện
xong.

Ý tưởng thực hiện

1. Thiết lập tên các câu lệnh , thành phần của từng câu lệnh, dạng lệnh (I,R,J) và chu kỳ của từng câu lệnh

2. Nhập câu lệnh muốn kiểm tra và lưu vào bộ nhớ

3. Tách từng thành phần của câu lệnh và lưu vào bộ nhớ: opcode, operand 1, operand 2, operand 3

4. Kiểm tra từng thành phần của câu lệnh, nếu sai ở thành phần nào thì chương trình sẽ dừng lại và không kiểm tra tiếp. Nếu đúng thì in ra dạng lệnh và số chu kỳ.

Khởi tạo dữ liệu ban đầu

-Type: chứa các thành phần của lệnh:
Trong đó:

- x: không chưa gì
- r: thanh ghi (register)
- i: số (immediately)
- f: thanh ghi số thực (float)
- l: nhãn dán (label)
- s: thành phần đặc biệt

```
type: .asciiz "x", "r", "i", "f", "l", "s"
register: .asciiz
        "$zero", "$at", "$v0", "$v1", "$a0", "$a1", "$a2", "$a3",
        "$t0", "$t1", "$t2", "$t3", "$t4", "$t5", "$t6", "$t7",
        "$s0", "$s1", "$s2", "$s3", "$s4", "$s5", "$s6", "$s7",
        "$t8", "$t9", "$k0", "$k1", "$gp", "$sp", "$fp", "$ra",
        "$0", "$1", "$2", "$3", "$4", "$5", "$6", "$7", "$8",
        "$9", "$10", "$11", "$12", "$13", "$14", "$15", "$16",
        "$17", "$18", "$19", "$20", "$21", "$22", "$23", "$24",
        "$25", "$26", "$27", "$28", "$29", "$30", "$31", "$32" ,"\0"
```

-Register: chứa tên của 32 thanh ghi và có được sắp xếp

```
float: .asciiz
        "$f0", "$f1", "$f2", "$f3", "$f4", "$f5", "$f6", "$f7",
        "$f8", "$f9", "$f10", "$f11", "$f12", "$f13", "$f14", "$f15",
        "$f16", "$f17", "$f18", "$f19", "$f20", "$f21", "$f22", "$f23",
        "$f24", "$f25", "$f26", "$f27", "$f28", "$f29", "$f30", "$f31", "\0"
```

-Float: chứa tên của các thanh ghi làm việc với số thực

Khởi tạo dữ liệu ban đầu

Instructions: chứa các lệnh được sắp xếp theo trình tự trong bảng chữ cái, các thành phần có trong lệnh, dạng lệnh và số chu kỳ thực hiện

```
instructions: .asciiz
        "abs.d","ffx"," ","1",
        "abs.s","ffx"," ","1",
        "add","rrr","R","1",
        "add.d","rrr"," ","1",
        "add.s","rrr"," ","1",
        "addi","rri","I","1",
        "addiu","rri","I","1",
        "addu","rrr","R","1",
        "and","rrr","R","1",
        "andi","rri","I","1",
        "bc1f","lxx"," ","2",
        "bc1t","lxx"," ","2",
        "beq","rrl","I","2",
        "bgez","rlx","I","2",
        "bgezal","rlx","I","2",
        "bgtz","rlx","I","2",
        "blez","rlx","I","2",
        "bltz","rlx","I","2",
        "bltzal","rlx","I","2",
        "bne","rrl","I","2",
        "break","xxx","R","2",
        "c.eq.d","ffx"," ","2",
        "c.eq.s","ffx"," ","2",
        "c.le.d","ffx"," ","2",
        "c.le.s","ffx"," ","2",
        "c.lt.d","ffx"," ","2",
        "c.lt.s","ffx"," ","2",
        "ceil.w.d","ffx"," ","1",
        "ceil.w.s","ffx"," ","1",
        "clo","rrx","R","1",
```

Khởi tạo dữ liệu ban đầu

Khởi tạo các ký tự đặc biệt và các câu dẫn dắt khi thực hiện chương trình

```
#character
tab     : "\t"
null    : "\0"
enter   : "\n"
space   : " "
comma   : ","
mark    : "'"
colon   : ":"

#say sth
Message:        .asciiz "Enter an Assembly command: "
Message1:       .asciiz "Type of Instruction: "
valid:          .asciiz "valid"
invalid:        .asciiz "invalid"
say_opcode:     .asciiz "Opcode "
say_operand:    .asciiz "Operand "
say_cycle:      .asciiz "Number of Clock Cycles: "
```

Thực hiện chương trình

Lấy ra opcode

```
opcode_take:
        addi $s1, $s1, 1
        sb $s2, 0($s1)
        addi $s0, $s0, 1
        lb $s2, 0($s0)
        lb $s3, 0($t0)
        beq $s3, $s2, end_take_opcode
        lb $s3, 0($t1)
        beq $s3, $s2, end_take_opcode
        lb $s3, 0($t2)
        beq $s3, $s2, end_take_opcode
        lb $s3, 0($t3)
        beq $s3, $s2, print_opcode
        j opcode_take
end_take_opcode:
        addi $s0, $s0, 1
        lb $s2, 0($s0)
        lb $s3, 0($t0)
        beq $s3, $s2, end_take_opcode
        lb $s3, 0($t3)
        beq $s3, $s2, end_take_opcode
        lb $s3, 0($t1)
        beq $s3, $s2, end_take_opcode
        lb $s3, 0($t2)
        beq $s3, $s2, end_take_opcode
        la $s4, opr1
        addi $s4, $s4, -1
```

Thực hiện chương trình

Lấy ra toán tử đầu tiên

```
opr1_take:
        addi $s4, $s4, 1
        sb $s2, 0($s4)
        addi $s0, $s0, 1
        lb $s2, 0($s0)
        lb $s3, 0($t0)
        beq $s3, $s2, end_take_opr1
        lb $s3, 0($t1)
        beq $s3, $s2, end_take_opr1
        lb $s3, 0($t2)
        beq $s3, $s2, end_take_opr1
        lb $s3, 0($t3)
        beq $s3, $s2, print_opcode
        j opr1_take
end_take_opr1:
        addi $s0, $s0, 1
        lb $s2, 0($s0)
        lb $s3, 0($t0)
        beq $s3, $s2, end_take_opr1
        lb $s3, 0($t3)
        beq $s3, $s2, end_take_opr1
        lb $s3, 0($t1)
        beq $s3, $s2, end_take_opr1
        lb $s3, 0($t2)
        beq $s3, $s2, end_take_opr1
        la $s4, opr2
        addi $s4, $s4, -1
```

Thực hiện chương trình

Lấy ra toán tử thứ 2

```
opr2_take:
        addi $s4, $s4, 1
        sb $s2, 0($s4)
        addi $s0, $s0, 1
        lb $s2, 0($s0)
        lb $s3, 0($t0)
        beq $s3, $s2, end_take_opr2
        lb $s3, 0($t1)
        beq $s3, $s2, end_take_opr2
        lb $s3, 0($t2)
        beq $s3, $s2, end_take_opr2
        lb $s3, 0($t3)
        beq $s3, $s2, print_opcode
        j opr2_take
end_take_opr2:
        addi $s0, $s0, 1
        lb $s2, 0($s0)
        lb $s3, 0($t0)
        beq $s3, $s2, end_take_opr2
        lb $s3, 0($t2)
        beq $s3, $s2, end_take_opr2
        lb $s3, 0($t3)
        beq $s3, $s2, end_take_opr2
        lb $s3, 0($t1)
        beq $s3, $s2, end_take_opr2
        la $s4, opr3
        addi $s4, $s4, -1
```

Thực hiện chương trình

Lấy ra toán tử cuối cùng và in ra opcode đã lấy ra

```
opr3_take:
        addi $s4, $s4, 1
        sb $s2, 0($s4)
        addi $s0, $s0, 1
        lb $s2, 0($s0)
        lb $s3, 0($t3)
        beq $s3, $s2, print_opcode
        j opr3_take
print_opcode:
        la $s0, instructions
        la $s1, opcode
        li      $v0, 4
        la      $a0, say_opcode
        syscall
        li      $v0, 4
        la      $a0, mark
        syscall
        li      $v0, 4
        la      $a0, opcode
        syscall
        li      $v0, 4
        la      $a0, mark
        syscall
        li      $v0, 4
        la      $a0, colon
        syscall
        li      $v0, 4
        la      $a0, space
        syscall
```

Thực hiện chương trình

Kiểm tra opcode đã lưu có hợp lệ hay không bằng cách kiểm tra xem nó có trong phần dữ liệu đã khai báo hay không?

```
check_opcode:
        lb $s2, 0($s0)
        lb $s3, 0($s1)
        bne $s2, $s3, next_opcode
        addi $s0, $s0, 1
        addi $s1, $s1, 1
        lb $s4, 0($s1)
        beqz $s4, opcode_endcheck
        j check_opcode
 next_opcode:
        beqz $s2, next_instruct
        addi $s0, $s0, 1
        lb $s2, 0($s0)
        j next_opcode
next_instruct:
        addi $s0, $s0, 9
        lb $s2, 0($s0)
        beqz $s2, invalid_opcode
        la $s1, opcode
        j check_opcode
opcode_endcheck:
        lb $t9, 0($s0)
        beqz $t9, valid_opcode
```

Thực hiện chương trình

Kiểm tra lần lượt 3 toán tử đã lưu xem có đúng với thành phần lệnh đã lưu khi khao báo hay không?

```
check_opr1:
        la $s2, opr1
        lb $t4, 0($s0)
        jal check_type
check_opr2:
        addi $s0, $s0, 1
        la $s2, opr2
        lb $t4, 0($s0)
        jal check_type
check_opr3:
        addi $s0, $s0,1
        la $s2, opr3
        lb $t4, 0($s0)
        jal check_type
        j print_type
```

Thực hiện chương trình

Kiểm tra lần lượt từng thành phần so với type

```
check_type:
        la $s4, register
        la $s3, type
        lb $t5, 0($s3)
        beq $t4, $t5, x_check
        li      $v0, 4
        la      $a0, say_operand
        syscall
        li      $v0, 4
        la      $a0, mark
        syscall
        li      $v0, 4
        addi    $a0, $s2, 0
        syscall
        li      $v0, 4
        la      $a0, mark
        syscall
        li      $v0, 4
        la      $a0, colon
        syscall
        li      $v0, 4
        la      $a0, space
        syscall
        addi $s3, $s3, 2
        lb $t5, 0($s3)
        beq $t4, $t5, check_r
        addi $s3, $s3, 2
        lb $t5, 0($s3)
        beq $t4, $t5,check_i
        addi $s3, $s3, 2
        lb $t5, 0($s3)
        beq $t4, $t5,check_f
        addi $s3, $s3, 2
        lb $t5, 0($s3)
        beq $t4, $t5,check_l
        addi $s3, $s3, 2
        lb $t5, 0($s3)
        la $s4, register
        beq $t4, $t5,check_s
```

Thực hiện chương trình

```
x_check:
        lb $t6, 0($s2)
        bnez $t6, invalid_x
        jr $ra
invalid_x:
        li      $v0, 4
        la      $a0, say_operand
        syscall
        li      $v0, 4
        la      $a0, mark
        syscall
        li      $v0, 4
        addi    $a0, $s2, 0
        syscall
        li      $v0, 4
        la      $a0, mark
        syscall
        li      $v0, 4
        la      $a0, colon
        syscall
        li      $v0, 4
        la      $a0, space
        syscall
```

Kiểm tra thành phần rỗng (x)

Thực hiện chương trình

```
check_r:
        la $s4, register
        add $s7, $s2, $0
        j r_loop
r_loop:
        lb $t5, 0($s4)
        lb $t6, 0($s7)
        bne $t5, $t6, r_next
        addi $s4, $s4, 1
        addi $s7, $s7, 1
        lb $t8, 0($s7)
        beqz $t8, return_valid
        j r_loop
r_next:
        addi $s4, $s4, 1
        lb $t7, 0($s4)
        bnez $t7, r_next
        addi $s4, $s4, 1
        lb $t7, 0($s4)
        beqz $t7, return_invalid
        add $s7, $s2, $0
        j r_loop
```

Kiểm tra thành phần thanh ghi (r)

Thực hiện chương trình

```
check_i:
        lb $t5, 0($s2)
        beq $t5, '-', i_next
        beq $t5, '+', i_next
i_loop:
        lb $t9, 0($s2)
        blt $t9, 0x30, return_invalid
        bgt $t9, 0x39, return_invalid
        addi $s2, $s2, 1
        lb $t9, ($s2)
        beqz $t9, return_valid
        j i_loop
i_next:
        addi $s2, $s2, 1
        j check_i
```

Kiểm tra thành phần số nguyên (i)

Thực hiện chương trình

```
check_f:
        la $s4, float
        add $s7, $s2, $0
        j f_loop
f_loop:
        lb $t5, 0($s4)
        lb $t6, 0($s7)
        bne $t5, $t6, f_next
        addi $s4, $s4, 1
        addi $s7, $s7, 1
        lb $t8, 0($s7)
        beqz $t8, return_valid
        j f_loop
f_next:
        addi $s4, $s4, 1
        lb $t7, 0($s4)
        bnez $t7, f_next
        addi $s4, $s4, 1
        lb $t7, 0($s4)
        beqz $t7, return_invalid
        add $s7, $s2, $0
        j f_loop
```

Kiểm tra thành phần thanh ghi số thực (f)

Thực hiện chương trình

```
check_1:
        lb $t5, 0($s2)
        addi $s2, $s2, 1
        blt $t5, 'A', return_invalid
        ble $t5,'Z', l_next
        blt $t5, 'a', return_invalid
        ble $t5,'z', l_next
l_loop:
        lb $t5, 0($s2)
        beq $t5, '_', i_next
        blt $t5,'A', return_invalid
        ble $t5, 'Z', l_next
        blt $t5,'a', return_invalid
        ble $t5, 'z',l_next
l_next:
        addi $s2, $s2, 1
        lb $t5, 0($s2)
        beqz $t5, return_valid
        j l_loop
```

Kiểm tra thành phần nhãn dán (l)

Thực hiện chương trình

```
check_s:
        lb $t5, 0($s2)
        beq $t5, '+', s_next
        beq $t5, '-', s_next
s_loopnumber:
        beq $t5, '(', s_rcheck
        blt $t5, '0', return_invalid
        bgt $t5, '9', return_invalid
s_next:
        addi $s2, $s2, 1
        lb $t5, 0($s2)
        beqz $t5, return_valid
        j s_loopnumber
s_rcheck:
         addi $s2, $s2, 1
         add $s7, $s2, $0
         j s_rloop
s_rloop:
        lb $t5, 0($s4)
        lb $t6, 0($s7)
        bne $t5, $t6, s_rnext
        addi $s4, $s4, 1
        addi $s7, $s7, 1
        lb $t8, 0($s7)
        beq $t8, ')', return_valid
        j s_rloop
```

```
s_rnext:
        addi $s4, $s4, 1
        lb $t7, 0($s4)
        bnez $t7, s_rnext
        addi $s4 , $s4 , 1
        lb $t7, 0($s4)
        beqz $t7, return_invalid
        add $s7, $s2, $0
        j s_rloop
```

Kiêm tra thành phần đặc biệt (s)

Thực hiện chương trình

```
print_type:
        addi $s0, $s0, 2
        lb $t5, 0($s0)
        beq $t5, ' ', clock_cycles
        li      $v0, 4
        la      $a0, Message1
        syscall
        li $v0, 11
        add  $a0, $t5, $0
        syscall
clock_cycles:
        addi $s0, $s0, 2
        lb $t5, 0($s0)
        li      $v0, 4
        la      $a0, enter
        syscall
        li      $v0, 4
        la      $a0, say_cycle
        syscall
        li $v0, 11
        add  $a0, $t5, $0
        syscall
        j endmain
```

In ra dạng lệnh và số chu kỳ thực hiện lệnh

Kết quả mô phỏng

Dữ liệu đầu vào

```
Enter an Assembly command: add $t2, $t3, $t4
```

Dữ liệu đầu ra

```
Enter an Assembly command: add $t2, $t3, $t4
Opcode 'add': valid
Operand '$t2': valid
Operand '$t3': valid
Operand '$t4': valid
Type of Instruction: R
Number of Clock Cycles: 1
-- program is finished running (dropped off bottom) --
```

Kết quả mô phỏng

Dữ liệu đầu vào

```
Enter an Assembly command: addi $t2, $t3, $f3
```

Dữ liệu đầu ra

```
Enter an Assembly command: addi $t2, $t3, $f3
Opcode 'addi': valid
Operand '$t2': valid
Operand '$t3': valid
Operand '$f3': invalid
-- program is finished running (dropped off bottom) --
```

Kết quả mô phỏng

Dữ liệu đầu vào

```
Enter an Assembly command: j vietanh
```

Dữ liệu đầu ra

```
Enter an Assembly command: j vietanh
Opcode 'j': valid
Operand 'vietanh': valid
Type of Instruction: J
Number of Clock Cycles: 2
-- program is finished running (dropped off bottom) --
```

Cảm ơn thầy và các bạn
đã lắng nghe!